PATENT **Docket No. RSW9-99-119**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

INVENTORS:	**Kenneth McClamroch** **Avi Yaeli** **Gobi Zodik**	Examiner: **C. Nguyen** Art Unit: **2161**
APPL. NO.	**09/473,554**	
FILED:	**December 29, 1999**	
CASE:	**RSW9-99-119**	
TITLE:	**ASSET LOCATOR**	

DECLARATION OF KENNETH MCCLAMROCH UNDER 37 CFR §1.131

I, Kenneth McClamroch, do hereby declare the following.

(1) I am one of the inventors of the invention claimed in the patent application assigned U.S. Serial No. 09/473,554 ("the present invention").

(2) I have read the currently pending claims of U.S. Serial No. 09/473,554.

(3) On information and belief, the present invention was scheduled for release on August 31, 1999 as part of the IBM product "VisualAge TeamConnection", Version/Release 3.0.3.

(4) The present invention was completed at least as early as its scheduled release date of August 31, 1999.

(5) On information and belief, the actual release date of "VisualAge TeamConnection", containing the present invention, was in September of 1999.

(6) The attached IBM disclosure document, entitled "Disclosure RSW8-1999-0171," was printed on October 22,, 1999 and was originated by me on August 9, 1999.

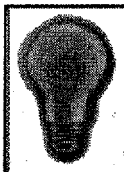
(7) Disclosure RSW8-1999-0171 provides a detailed disclosure of the present invention as currently claimed in U.S. Serial No. 09/473,554. Page 6 of the Disclosure RSW8-1999-0171 refers to the scheduled release of VisualAge TeamConnection, containing the present invention, on August 31, 1999.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: 4/25/06

Kenneth R McClamroch
Ken McClamroch

S:\IBM\IBM Raleigh RSW\Patents\P23546 USA\PTO\Declaration of Ken McClamroch.doc



Disclosure RSW8-1999-0171

Created By: Ken McClamroch **Created On:** 08/09/99 01:55:04 PM
Last Modified By: wpts1 wpts1 **Last Modified On:** 10/07/99 02:18:59 PM

*** IBM Confidential ***

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

Summary

Status	Final Decision **(File / N/A)**
Docket Family	RSW9-1999-0119
Processing Location	RSW
Functional Area	Greg Clark
Attorney/Patent Professional	Gregory Doudnikoff/Raleigh/IBM
IDT Team	Steven Miller/Raleigh/IBM; Art Francis/Raleigh/IBM; David Kuehr-Mclaren/Raleigh/IBM; Richard Gray/Raleigh/IBM; Allan K Edwards/Raleigh/IBM; Sandeep Singhal/Raleigh/IBM; Mark Peters/Raleigh/IBM; R Redpath/Raleigh/IBM; Scott Rich/Raleigh/IBM
Submitted Date	08/09/99 04:21:27 PM
Owning Division	SWSD Add/Change
PVT Score	To calculate a PVT score, use the 'Calculate PVT' button.
Lab	
Technology Code	
Incentive Program	

Inventors with Lotus Notes IDs

Inventors: Gabi Zodik/Haifa/IBM, Avi Yaeli/Haifa/IBM, Ken McClamroch/Raleigh/IBM

Inventor Name > denotes primary contact	Inventor Serial	Div/Dept	Manager Serial	Manager Name
Zodik, Gabi	958606	N/A/403	145307	Bernstein, David D
Yaeli, Avi	755409	N/A/403	958606	Zodik, Gabi
McClamroch, K L (Ken)	611091	76/TJOB	011060	Allen, Pam

Inventors without Lotus Notes IDs

IDT Selection

IDT Team:	Attorney/Patent Professional:
Steven Miller/Raleigh/IBM Art Francis/Raleigh/IBM David Kuehr-Mclaren/Raleigh/IBM Richard Gray/Raleigh/IBM Allan K Edwards/Raleigh/IBM Sandeep Singhal/Raleigh/IBM Mark Peters/Raleigh/IBM R Redpath/Raleigh/IBM Scott Rich/Raleigh/IBM	Gregory Doudnikoff/Raleigh/IBM

Response Due to IP&L : 09/12/99

Main Idea

***Title of disclosure (in English)**

Asset Locator

***Idea of disclosure**

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

This invention (to be referred to as Asset Locator in the remainder of the disclosure) provides the capability to gather information about assets contained in repository(ies) and capture this information in a database for subsequent searches. You can use Asset Locator to capture information from across your enterprise and consolidate in a single database for which multiple users can access. Users can then perform the following types of queries against this stored information:

- Free-text search: "search engine" style queries looking for a keyword or phrase across all sources.
- Attribute-specific search: The ability to invoke queries on specific source attributes. For example, "search for all classes that are a subclass of class HashTable" or "search all classes that implement method foo()". The list of searchable attributes will vary by programming language.
- The ability to mix both free-text queries with attribute-specific queries. For example, "search for all classes that are a subclass of class HashTable" AND contain the phrase "initial capacity". This ability to mix free-text with semantic query options is the key Asset Locator differentiator. Here's a more detailed example to illustrate the capabilities of Asset Locator:

  Suppose one is searching for a class to implement a quick sort algorithm. The user has no idea the name of the class for which he is searching. So he starts by invoking a mixed query where the free text includes several keywords, like : **"sort" "quick" "descending"**. In addition, he states that the **classname** attribute should include the string **"*sort*"** this is done in order to narrow down results, he also believes that at least one **method** should include the name **"*sort*** " so the methods attributes has also a value. Performing this query will result in a list of classes from which in most cases (if the repository includes any quick sort classes at all) the quick sort class will be ranked with the highest value while less relevant classes with ranks of lower value.

There are multiple components that enable this functionality:

- **Crawler**

The Crawler has two primary roles:

- Interfacing to the repositories (e.g., TeamConnection) and extracting the desired sources
- Invoking the suitable analyzer against each of the source files encountered

The Crawler includes a scheduler that is responsible for starting the crawling processes, allowing potentially time intensive operations to be performed when resources are idle. For example, I can have my repositories crawled by Asset Locator every morning at 2am.

- **Analyzers**

There are a set of analyzers that are invoked by the Crawler based on the types of sources encountered. These analyzers are responsible for parsing the various information sources. There are currently analyzers to support the following source types:

- Java
- C/C++

- COBOL
- HTML
- XML

All the analyzers, for each of the supported types, extract the file path and the last update time for the resource. In addition, each analyzer constructs a free-text attribute based on resource comments and other textual information as well as structural information which varies by source type.

Let's take the C/C++ analyzer as an example:
When the Crawler encounters a C or C++ resource, the C/C++ analyzer is invoked and captures the following information:

- File path
- Free text constructed out of: all types of comments, class name, base class name(s), method names, and class member variable names
- Class name(s)
- Base class(es)
- Class member variables
- Class methods
- Include files
- Templates
- Last update date/time

In addition to the default set of analyzers, Asset Locator will provide a framework allowing user-defined analyzers to be "plugged-in" as well as the ability for customization of the current set of analyzers to include end-user specific information (e.g., customized code metrics based on specific development environment).
This framework allows the user to implement their own analyzer for the desired file types and register this analyzer with Asset Locator via the configuration file. Asset Locator is then responsible for dynamically loading this analyzer when a suitable resource is encountered.

- **Runtime servlet**

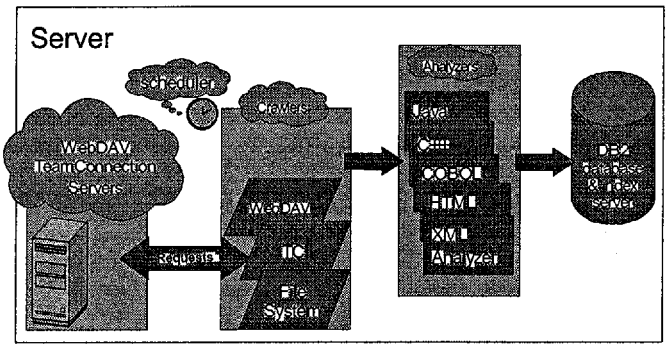
Once the repository data is crawled and captured in the Asset Locator database, queries can be performed. The runtime servlet is responsible for processing these queries and interacting with the database to return the appropriate results based on the specified query. The servlet can return results in either HTML (to be displayed in a browser, for example) or as an XML stream.

- **User Interface to initiate queries/display results**

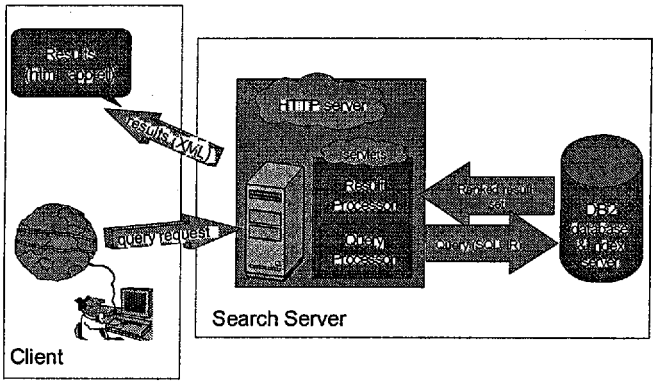
The user interface allows users to specify queries of varying complexity and generates the HTTP request to invoke the Asset Locator servlet. The servlet returns the results in an XML stream which is parsed by the user interface and formatted for presentation to the end user. The user can then perform library management actions against these results, including viewing the contents, extracting, checking in/out of the selected assets.
Asset Locator is designed to run as a daemon which requires no management or maintenance once configured.

The following are a couple of diagrams to illustrate Asset Locator crawler and runtime architectures:

Information Gathering Architecture



Run Time Architecture



2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

The ability to both capture and then perform searches against your repository assets provides many uses, including:

- Reuse

Using Asset Locator, users will have the ability to quickly identify components that are candidates for re-use. Not only can they identify such re-usable components, but they can also use Asset Locator to determine which assets may be using these components today. I'll use our development environment to demonstrate.

A portion of our development project is developed using Java. Java has a notion of packaging to allow grouping of related Java sources together. One of our development packages is **com.ibm.project.reuse** which is used to group Java classes that are re-usable. This package includes, for example, a notebook control, toolbar control, button classes, etc.

Asset Locator provides the ability to perform both free-text queries against all sources and attribute specific queries for all sources of a particular language. For Java sources, one of the attributes you can query against is **package** name. So if you perform an **Asset Locator query** against all Java sources with package name **com.ibm.project.reuse***, you will receive results listing all Java sources included in that package. **So with a very simple query, you have just identified all of our Java classes intended for re-use.**

Now you may want to know who is using these re-usable components today to gain a better understanding of how others have been able to use these classes. In Java, you use a package by specifying an **import** statement in the Java source that will use classes from that package (e.g., **import com.ibm.project.reuse.***) Asset Locator allows you to query all Java sources based on the **import** attribute. So with an **Asset Locator query** against all Java sources that **import** package **com.ibm.project.reuse***, you **very quickly identified all assets that are using re-usable components from the reuse package.**

Of course, development groups will have their unique conventions for identifying reusable components. Since Asset Locator supports free-text queries that are very similar to search engines we are all familiar with, there is a great deal of flexibility to allow support of a wide range of conventions.

- Impact Analysis

Another key use of Asset Locator is impact analysis, which is the ability to determine the impact a change will make on all parts in your library. A classic example of such a change that caused a major impact in the Java programming community was when Sun decided to change all their Java user interface package names from **com.sun.java.swing.*** to **javax.swing.***. This change required all code currently using these Java packages to change to use the new package name **javax.swing**. With Asset Locator, you can quickly determine the parts that would be affected by such a change by specifying an **Asset Locator query** against all Java sources that import package **com.sun.java.swing***.

Asset Locator quickly identified all parts that are impacted by such a change and allowed you to very easily perform the changes necessary.

- Search & Discovery

This value point is really more generic than the specific cases of reuse and impact analysis. Search & Discovery addresses the "search-engine style" interface that allows you to very quickly search for keywords or phrases to try and discover which assets might perform tasks you are interested in. For example, I may be interested in which sources (Java, C, C++, etc.) might employ hashtable technology. Using Asset Locator, I can specify a free-text query against **sources of all types** for the keyword **hashtable.** Unlike the very specific queries against only Java sources in the previous examples, this query will return any reference to hashtable in any supported source. The currently supported source types are Java, C/C++, COBOL, HTML, XML. There may be examples of hashtable usage in Java or C++. There may be HTML content discussing hashtable usage, etc.

Using Asset Locator in this manner is a definite productivity boost to quickly identify across your entire enterprise potential areas of interest. One of the primary advantages of Asset Locator is the ability to issue very specific queries against both free-text and structural attribute specific information to narrow your search criteria and receive the most relevant results.

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have

those others solved it and does your solution differ and why is it better?

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.

Asset Locator is a component of Visual Age Team Connection v3.0.3 with scheduled availability of 3Q99.

REDACTED

***Critical Questions (Questions 1 - 7 must be answered)**

***Question 1** On what date was the invention workable? [redacted] **Please format the date as MM/DD/YYYY** (Workable means i.e. when you know that your design will solve the problem)

***Question 2** Is there any planned or actual publication or disclosure of your invention to anyone outside IBM?	● Yes ○ No
If yes, Enter the name of each publication or patent and the date published below. Publication/Patent: VisualAge TeamConnection Administration Guide - v3.0.3 Date Published or Issued: 08/31/1999 (via TeamConnection v3.0.3 availability)	
Are you aware of any publications, products or patents that relate to this invention?	○ Yes ● No
If yes, Enter the name of each publication or patent and the date published below. Publication/Patent: Date Published or Issued:	

***Question 3** Has the subject matter of the invention or a product incorporating the invention been sold, used internally in manufacturing, announced for sale, or included in a proposal?	● Yes ○ No
Is a sale, use in manufacturing, product announcement, or proposal planned?	● Yes ○ No
If Yes, identify the product if known and indicate the date or planned date of sale, announcements, or proposal and to whom the sale, announcement or proposal has been or will be made. Product: VisualAge TeamConnection Version/Release: 3.0.3 Code Name: Date: 08/31/1999 To Whom: General availability If more than one, use cut and paste and append as necessary in the field provided.	

***Question 4** Was the subject matter of your invention or a product incorporating your invention used in public, e.g., outside IBM or in the presence of non-IBMers?	○ Yes ● No
If yes, give a date. **Please format the date as MM/DD/YYYY**	

REDACTED